# Optimizing the Constant-$Q$ Transform in Octave

Hans Fugal

Department of Computer Science
New Mexico State University


Linux Audio Conference, 2009

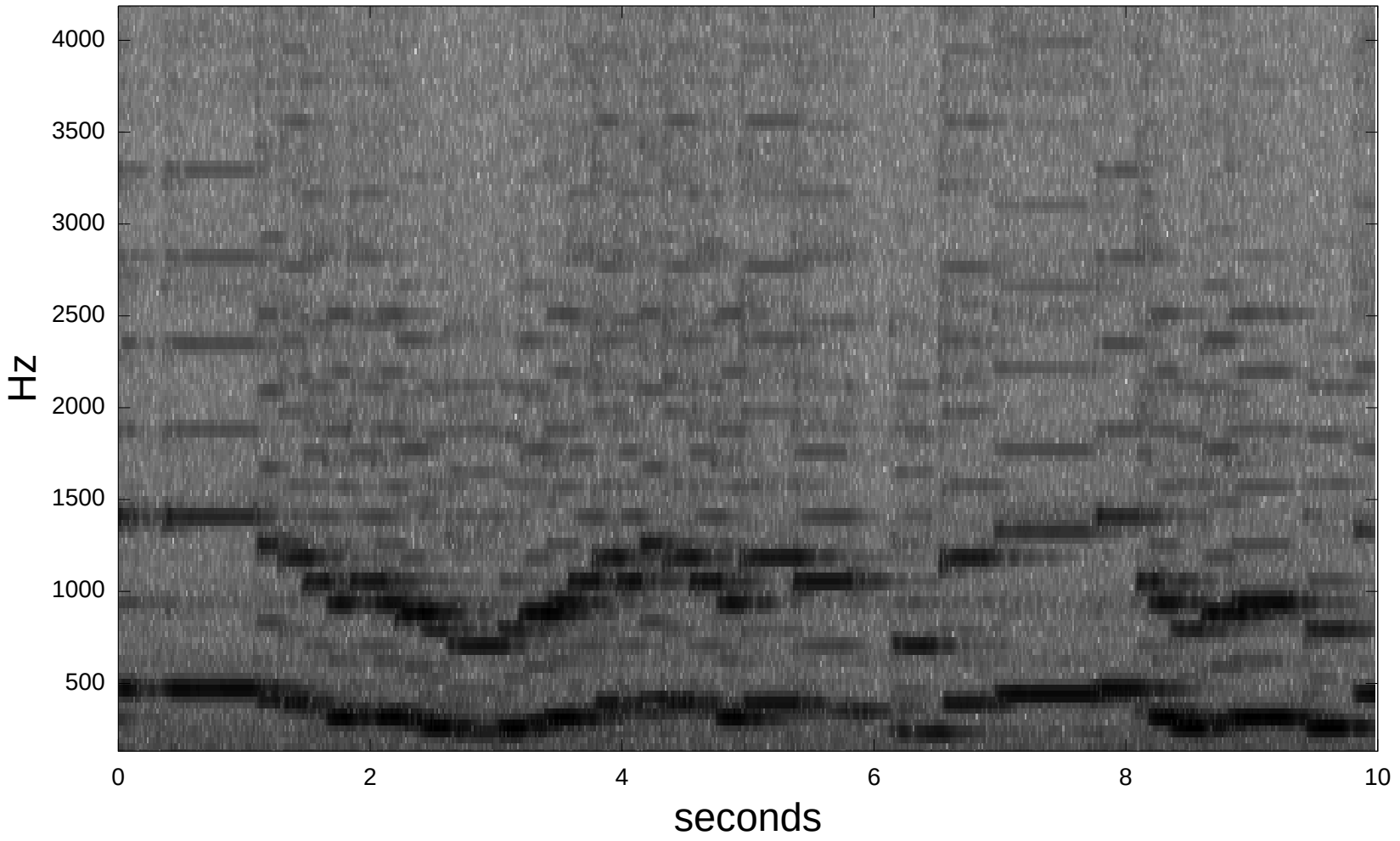
# Outline

# Octave

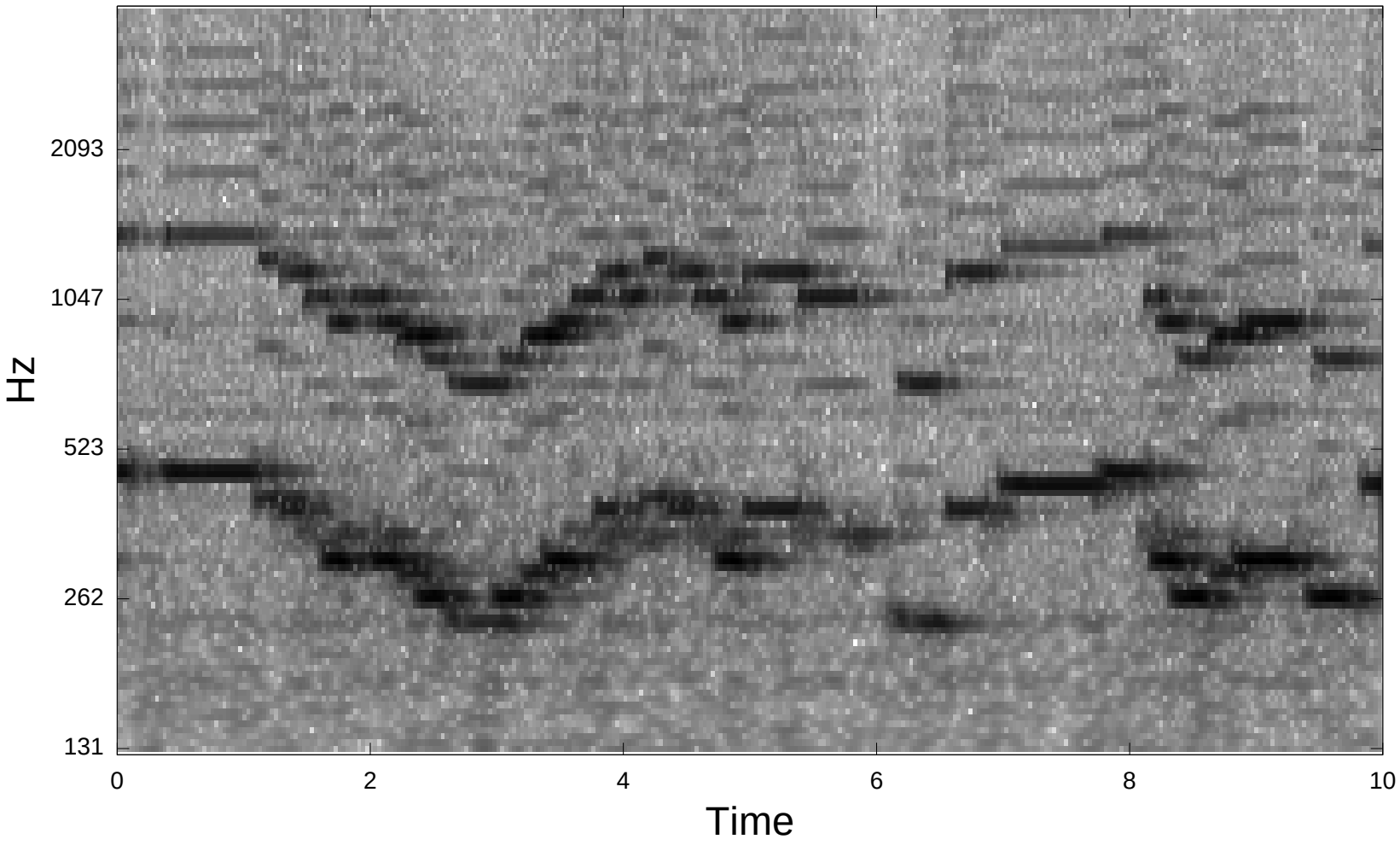
- `http://www.gnu.org/software/octave/`
- High-level language for numerical computations
- Mostly compatible with MATLAB (`http://www.mathworks.com/`)

> *Just about everyone thinks that the name Octave has something to do with music, but it is actually the name of one of the author's former professors who wrote a famous textbook on chemical reaction engineering, and who was also well known for his ability to do quick "back of the envelope" calculations. We hope that this software will make it possible for many people to do more ambitious computations just as easily.*

# Linear-frequency Spectrogram



# Log-frequency Spectrogram



# Constant-$Q$ Transform

- [Brown, 1991]
- One way to get a log-frequency spectrum
- $Q = f/\Delta f$, frequency over bandwidth
  - DFT has constant bandwidth, so $Q$ changes with frequency
  - Constant-$Q$ transform varies bandwidth with frequency
- Bandwidth is inversely proportional to number of samples

# Constant-$Q$ Transform Definition

$$X_{\mathrm{cq}}(k_{\mathrm{cq}}) = \frac{1}{N_{k_{\mathrm{cq}}}} \sum_{n=0}^{N_{k_{\mathrm{cq}}}-1} w_{k_{\mathrm{cq}}}(n) x(n) e^{-j2\pi Q n / N_{k_{\mathrm{cq}}}}$$

- $Q = 1/(2^{1/b} - 1)$ with $b$ bins per octave (12 for semitone spacing)
- $f_{k_{\mathrm{cq}}} = 2^{k_{\mathrm{cq}}/b} f_{\mathrm{min}}$ is the center frequency of the $k_{\mathrm{cq}}$th bin.
- $N_{k_{\mathrm{cq}}} = Q f_s / f_{k_{\mathrm{cq}}}$ is the number of samples
- $w_{k_{\mathrm{cq}}}$ is a window function (e.g. Hanning)

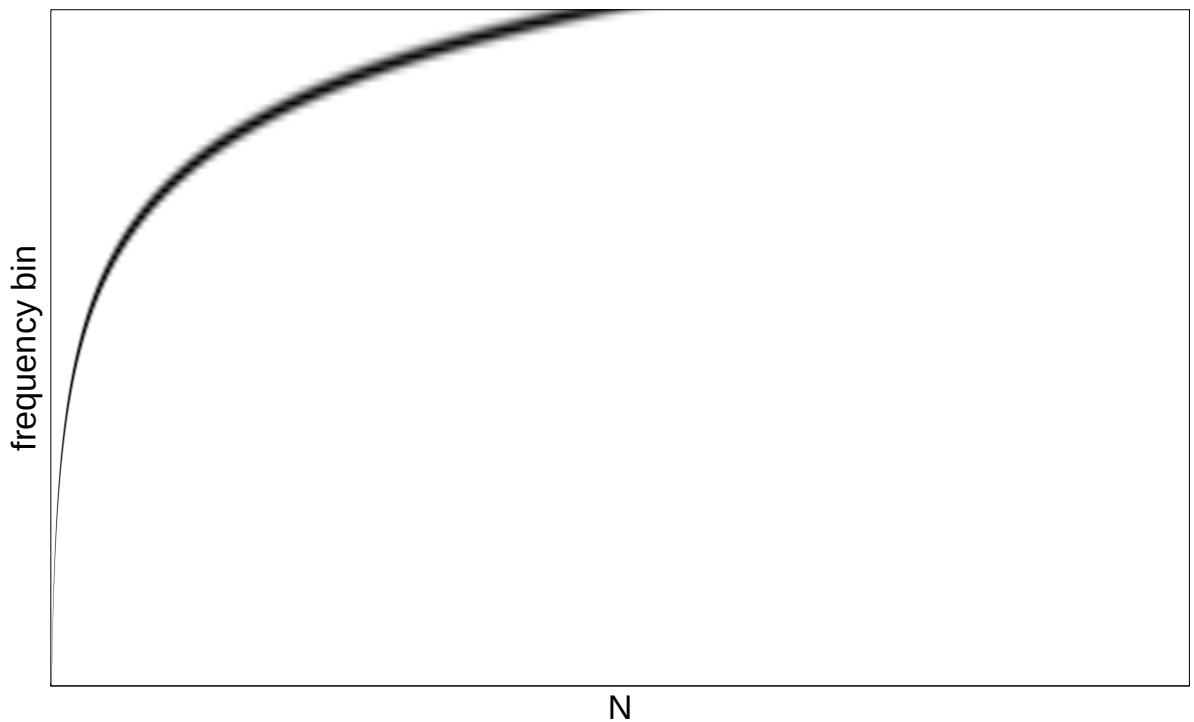
# Brown-Puckette Algorithm

- [Brown and Puckette, 1992]
- Can be more efficient to compute
- FFT $\times$ (sparse) spectral kernel

$$X_{\text{cq}}(k_{\text{cq}}) = \frac{1}{N_0} \sum_{k=0}^{N_0-1} X(k) K(k_{\text{cq}}, k)$$

# Spectral Kernel

$$K(k_{cq}, k) = \sum_{n=0}^{N_0-1} w_{k_{cq}}(n) e^{j2\pi Q/N_{k_{cq}}} e^{-j2\pi kn/N_0}$$



# Code

Download code at
`http://hans.fugal.net/research/cq-octave/`

| | |
|---|---|
| `cq` | Original |
| `ecq` | Brown-Puckette |
| `vecq` | Vectorized Brown-Puckette |
| `secq` | Brown-Puckette with sparse kernel |
| `svecq` | Vectorized Brown-Puckette with sparse kernel |
| `cqkern` | Spectral kernel |
| `sparse` | Convert spectral kernel to sparse matrix |

# cq

$$X_{\mathrm{cq}}(k_{\mathrm{cq}}) = \frac{1}{N_{k_{\mathrm{cq}}}} \sum_{n=0}^{N_{k_{\mathrm{cq}}}-1} w_{k_{\mathrm{cq}}}(n) x(n) e^{-j2\pi Qn/N_{k_{\mathrm{cq}}}}$$

```
1  for kcq = 0:B-1
2    f = f_min * r^kcq;
3    Nkcq = round(Q*sr/f);
4    Xcq(kcq+1) = sum(hamming(Nkcq) .* (x(1:Nkcq) \
5      .* exp(j2piQn(1:Nkcq)/Nkcq))) / Nkcq;
6  end
```

# ecq

$$X_{\mathrm{cq}}(k_{\mathrm{cq}}) = \frac{1}{N_0} \sum_{k=0}^{N_0-1} X(k) K(k_{\mathrm{cq}}, k)$$

```
1  X = fft(x, N0);
2  for kcq = 1:B
3    Xcq(kcq) = sum(X .* K(kcq, :)) / N0;
4  end
```

# vecq

$$X_{\mathrm{cq}}(k_{\mathrm{cq}}) = \frac{1}{N_0} \sum_{k=0}^{N_0-1} X(k) K(k_{\mathrm{cq}}, k)$$

```
1  X = fft(x, N0);
2  Xcq = sum((repmat(X, B, 1) .* K), 2) / N0
```

# Parameters—First Experiment

- $f_s = 44100\,\text{Hz}$
- $b = 12$ bins per octave (semitone spacing)
- $f_{\text{min}} = 16.35\,\text{Hz}$ (C0)
- $f_{\text{max}} = 22050\,\text{Hz}$ (Nyquist frequency)

# Parameters—Second Experiment

- $f_s = 44100\,\text{Hz}$
- $b = 12$ bins per octave (semitone spacing)
- $f_{\text{min}} = 130.81\,\text{Hz}$ (C3)
- $f_{\text{max}} = 22050\,\text{Hz}$ (Nyquist frequency)

# Specifications

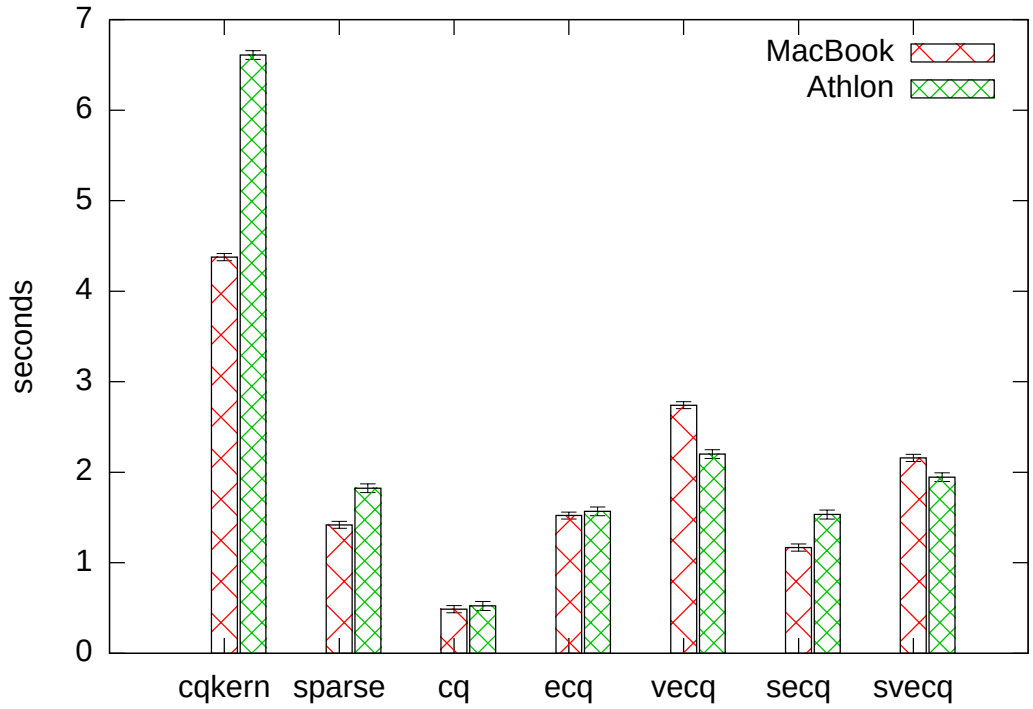
MacBook

- 2 GHz Intel Core Duo
- 2 GB RAM
- OS X 10.5 (Leopard)
- Octave 3.0.3 from MacPorts

Athlon

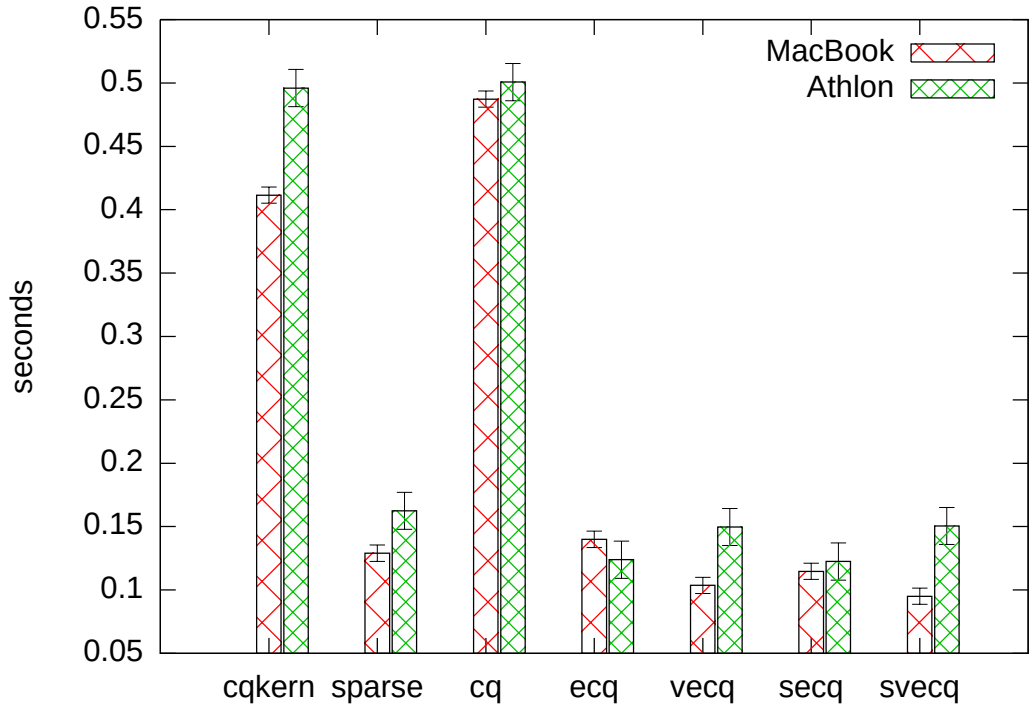
- Athlon 64 2800+ (1.8 GHz)
- 1 GB RAM
- 32-bit Ubuntu 8.10
- Octave 3.0.1 stock

# Results, $f_{\min} = C0$




- cq is fastest
- Vectorized are slower
- Sparse is faster

# Results, $f_{\min} = C3$




- cq is slowest
- Vectorized depends on hardware
- Sparse depends on hardware

# Discussion

- Brown-Puckette algorithm not always more efficient
  - Same order: $O(N_0 \log N_0)$
  - More space: $O(N_0 \log N_0)$ vs. $O(N_0)$
- Vectorization worse or marginal, probably not worth it
- Sparse marginal improvement, may be worth it

# Future Work

- MATLAB
  - relative and absolute comparison
  - `repmat`
- C
  - fast loops
  - more control over memory

# References


Judith C. Brown. Calculation of a constant Q spectral transform. *J. Acoust. Soc. Am*, 89(1):425–434, January 1991.

Judith. C. Brown and Miller S. Puckette. An efficient algorithm for the calculation of a constant Q transform. *J. Acoust. Soc. Am*, 92(5):2698–2701, November 1992.